

Resumen Capítulo 2: trata acerca del pre-procesamiento de los datos, análisis exploratorio de datos y la construcción de los modelos predictivos.

2.1. Descripción del problema y objetivos:

Altas concentraciones de ciertas algas nocivas en los ríos, constituyen un serio problema ecológico con un fuerte impacto no solo en las formas de vida de los ríos, sino también en la calidad del agua. Ser capaz de monitorear y realizar un pronóstico temprano de las algas, es esencial para mejorar la calidad de los ríos.

Varias muestras de agua fueron recolectadas en diferentes ríos Europeos, así como en varias fechas durante un periodo aproximadamente de un año. Por cada muestra de agua, diferentes propiedades químicas fueron medidas, así como también la frecuencia de ocurrencia de las siete algas nocivas. Otras características del proceso de recolección de agua también fueron registradas, tales como, la estación del año, el tamaño del río, y la velocidad del caudal del río.

Una de las motivaciones principales detrás de esta aplicación yace en el hecho que el monitoreo químico es económico y fácilmente automatizado, mientras el análisis biológico de las muestras para identificar el tipo de alga que está presente en el agua involucra examinación microscópica y requiere mano de obra, siendo ambas costosas y demandando una gran cantidad de tiempo. De esta manera, obtener modelos que predigan con precisión la frecuencia de las algas basado en las propiedades químicas facilitaría la creación de un sistema económico y automatizado para monitorear las algas nocivas.

Otro objetivo del estudio es proveer un mejor entendimiento de los factores que influyen en la frecuencia de las algas. Se quiere entender como estas frecuencias están relacionadas con ciertos atributos químicos de las muestras de agua, así como otras características, entre las que se encuentran estación del año, tipo de río, entre otras.

2.2. Descripción de los datos.

Los datos disponibles para este problema fueron recolectados en el contexto de la Red de Investigación **EURODIT** y utilizados en la competencia internacional de análisis de datos **COIL 1999**.

Hay dos conjuntos de datos principales para este problema. El primero consiste de datos para las 200 muestras de agua. Para ser más precisos, cada observación en el conjunto de datos disponibles es en efecto, un agregado de varias muestras de agua recolectadas del mismo río por un periodo de tres meses, durante la misma estación del año.

Cada observación contiene información acerca de las 11 variables. Tres de estas variables son nominales, y describen la estación del año cuando las muestras de agua a ser agregadas fueron recolectadas, así como el tamaño y la velocidad del caudal del río. Las ocho variables restantes son valores de diferentes parámetros químicos medidos en las muestras de agua formando el agregado, las cuales son:

- Valor de pH máximo.
- Mínimo valor de O_2 (Oxígeno)
- Valor medio de Cl (Cloro)
- Valor medio de NO_3^- (Nitrato)
- Valor medio de NH_4^+ (Amonio)
- Media de PO_3^{4-} (Ortofosfato)
- Media del total PO_4 (Fosfato)
- Media del Clorofill

Asociados a cada uno de estos siete parámetros hay siete números de frecuencias de diferentes algas contaminantes Encontradas en las respectivas muestras de agua. Ninguna información es dada respecto a los nombres de las algas que fueron identificadas.

El segundo conjunto de dato contiene información sobre 140 observaciones extras. Este usa la misma estructura básica pero no incluye información concerniente a la frecuencia de las siete algas contaminantes. Estas observaciones pueden permanecer como un conjunto de datos de prueba. La meta principal del estudio es predecir las frecuencias de las siete algas para las 140 muestras de agua. Esto significa que es una tarea de minería de datos de carácter predictivo. En este tipo de tarea, la meta principal es obtener un modelo que nos lleve a predecir el valor de una cierta variable objetivo dados los valores de un conjunto de variables predictoras. Este modelo puede también proveer indicaciones de cuales variables predictoras tienen mayor impacto sobre la variable objetivo.

2.3. Cargar los datos en R.

La función `head()` muestra las primeras seis líneas de cualquier data frame.

Hay tres archivos disponibles en la sección “Data” de la página web del libro. Los datos de “entrenamiento”, el cual contiene las 200 muestras de agua se encuentran en el archivo llamado “Analysis.txt” los datos de “prueba” se encuentran en “Eval.txt”, el cual es un archivo que contiene las 140 muestras de prueba. Existe un archivo adicional llamado “Sols.txt” que contiene la frecuencia de las algas de las 140 muestras de prueba. Este último archivo será utilizado para verificar el desempeño de los modelos predictivos y se tomarán como información desconocida por ahora. Cada línea de los archivos de entrenamiento y de prueba contienen los valores de las variables (de acuerdo a la descripción dada en la sección 2.2) separada por espacios. Los valores desconocidos o faltantes, son indicados con los caracteres “XXXXXXX”.

Lo primero que se debe hacer es descargar los tres archivos de la página web del libro y almacenarlos en algún directorio del disco duro (preferiblemente en el directorio de trabajo actual donde se corre la sesión de R, el cual se puede verificar a través del comando `getwd()` en el terminal).

Después de descargar los tres archivos en un directorio local, se debe comenzar por cargar en R los datos del archivo “Analysis.txt” (que son los datos de entrenamiento, los cuales serán utilizados

para obtener los modelos predictivos). Para leer los datos del archivo es suficiente ejecutar el siguiente comando:

```
> algae <- read.table('Analysis.txt',  
+                     header=F,  
+                     dec='.',  
+                     col.names=c('season','size','speed','mxPH','mnO2','Cl',  
+                                 'NO3','NH4','oPO4','PO4','Chla','a1','a2','a3','a4',  
+                                 'a5','a6','a7'),  
+                     na.strings=c('XXXXXXX'))
```

- El parámetro **header=F**, indica que el archivo a leer no incluye una primera línea con el nombre de las variables.
- **dec='.'** denotan que los números utilizan el carácter '.' para separar la cifra entera de la parte decimal.
Estos dos parámetros previos se pudieron haber omitido, ya que se está utilizando los valores por defecto.
- **col.names** para proveer un vector con los nombres que se le van a dar a las variables cuyos valores están siendo leídos.
- Finalmente, **na.strings** sirve para indicar un vector de caracteres que son interpretados como valores desconocidos. Estos valores son representados internamente en R por el valor NA.

2.4. Visualización de los datos y resúmenes.

Para tener una mejor idea del problema es recomendable investigar algunas propiedades estadísticas de los datos, y es siempre una buena idea comenzar el análisis con algún tipo de análisis exploratorio a los datos. Una primera impresión de las propiedades estadísticas de los datos se puede obtener a través de los resúmenes de los estadísticos descriptivos.

Dichos estadísticos se muestran a través del siguiente comando:

```
> summary(algae)
```

season	size	speed	mxPH	mnO2
autumn:40	large :45	high :84	Min. :5.600	Min. : 1.500
spring:53	medium:84	low :33	1st Qu.:7.700	1st Qu.: 7.725
summer:45	small :71	medium:83	Median :8.060	Median : 9.800
winter:62			Mean :8.012	Mean : 9.118
			3rd Qu.:8.400	3rd Qu.:10.800
			Max. :9.700	Max. :13.400
			NA's :1.000	NA's : 2.000

C1	N03	NH4	oP04
Min. : 0.222	Min. : 0.050	Min. : 5.00	Min. : 1.00
1st Qu.: 10.981	1st Qu.: 1.296	1st Qu.: 38.33	1st Qu.: 15.70
Median : 32.730	Median : 2.675	Median : 103.17	Median : 40.15
Mean : 43.636	Mean : 3.282	Mean : 501.30	Mean : 73.59
3rd Qu.: 57.824	3rd Qu.: 4.446	3rd Qu.: 226.95	3rd Qu.: 99.33
Max. :391.500	Max. :45.650	Max. :24064.00	Max. :564.60
NA's : 10.000	NA's : 2.000	NA's : 2.00	NA's : 2.00

P04	Chla	a1	a2
Min. : 1.00	Min. : 0.200	Min. : 0.00	Min. : 0.000
1st Qu.: 41.38	1st Qu.: 2.000	1st Qu.: 1.50	1st Qu.: 0.000
Median :103.29	Median : 5.475	Median : 6.95	Median : 3.000
Mean :137.88	Mean : 13.971	Mean :16.92	Mean : 7.458
3rd Qu.:213.75	3rd Qu.: 18.308	3rd Qu.:24.80	3rd Qu.:11.375
Max. :771.60	Max. :110.456	Max. :89.80	Max. :72.600
NA's : 2.00	NA's : 12.000		

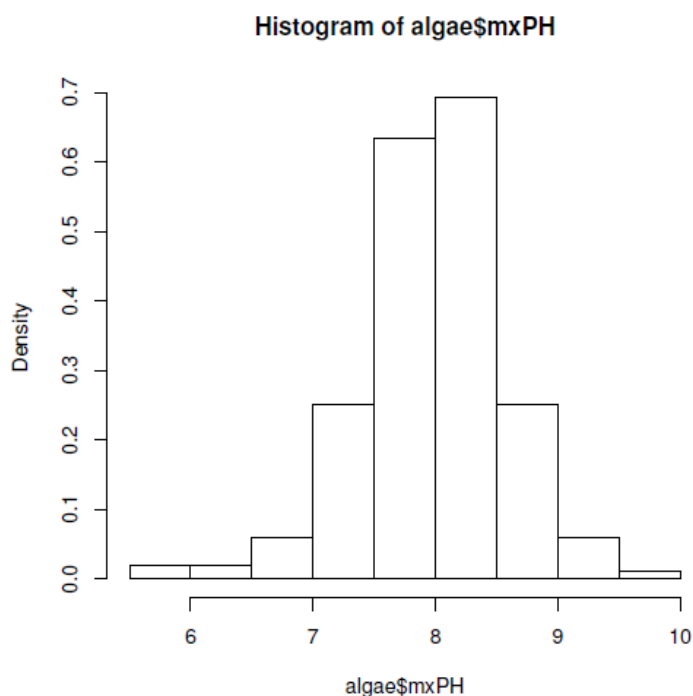
a3	a4	a5	a6
Min. : 0.000	Min. : 0.000	Min. : 0.000	Min. : 0.000
1st Qu.: 0.000	1st Qu.: 0.000	1st Qu.: 0.000	1st Qu.: 0.000
Median : 1.550	Median : 0.000	Median : 1.900	Median : 0.000
Mean : 4.309	Mean : 1.992	Mean : 5.064	Mean : 5.964
3rd Qu.: 4.925	3rd Qu.: 2.400	3rd Qu.: 7.500	3rd Qu.: 6.925
Max. :42.800	Max. :44.600	Max. :44.400	Max. :77.600

a7
Min. : 0.000
1st Qu.: 0.000
Median : 1.000
Mean : 2.495
3rd Qu.: 2.400
Max. :31.600

- En el caso de las variables cualitativas o nominales este da la frecuencia para cada posible valor. Por ejemplo, se puede observar que hay más muestras de agua recolectadas en invierno (Winter) que en alguna otra estación.
- Para las variables cuantitativas o numéricas, R nos da una serie de estadísticos tales como, la media, la mediana, información de los cuartiles y los valores extremos. Estos estadísticos proveen una primera idea de la distribución de los valores de las variables. Cuando alguna variable presenta valores nulos o desconocidos, dicha cantidad es representada a través de los caracteres NA's. Observando la diferencia entre la mediana y la media, así como el rango inter-cuartílico (3er cuartil menos el 1er cuartil), se puede obtener una idea de la asimetría de la distribución y también su dispersión. La mayoría de las veces, esta información se puede observar de manera gráfica. Como se mostrará a continuación mediante el comando:

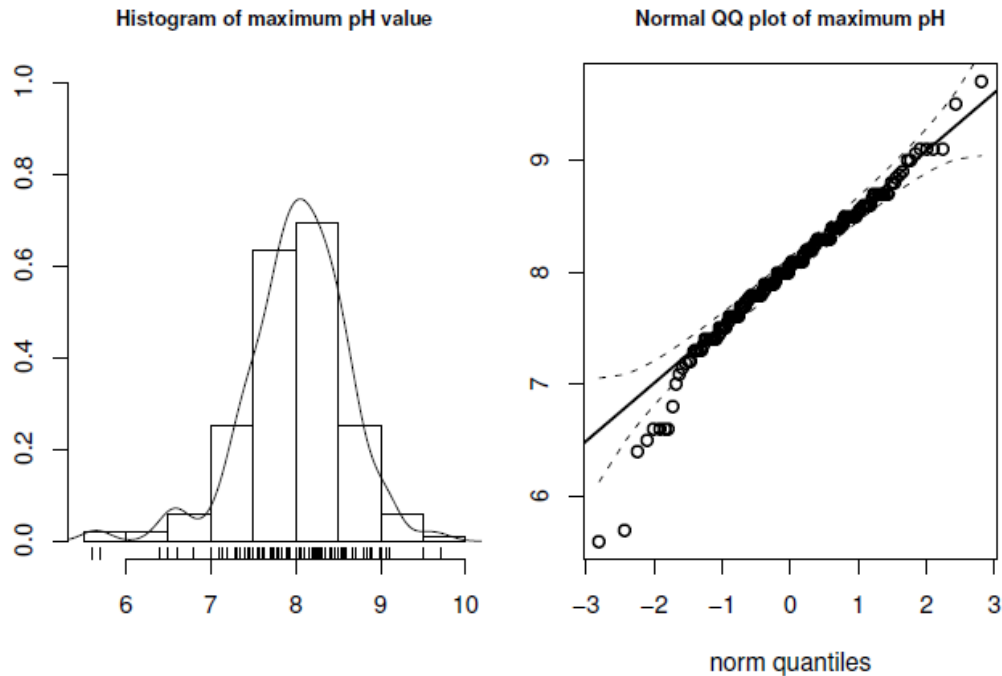
```
> hist(algae$mxPH, prob = T)
```

La instrucción anterior muestra el histograma de la variable **mxPH**. El resultado se muestra a continuación. Con el parámetro **prob = T** se obtienen las probabilidades de cada intervalo de valores.



La figura muestra que los valores de la variable **mxPH** aparentemente siguen una distribución muy cercana a la normal, con valores muy cercanos alrededor de la media. Una verificación más precisa de esta hipótesis se puede obtener a través del gráfico normal Q-Q. la función **qq.plot()**, en el

paquete `car()`, obtiene este tipo de gráfico, el resultado se muestra en la siguiente figura, junto a una versión más sofisticada de mostrar el histograma.



El gráfico anterior se obtuvo a través del siguiente código:

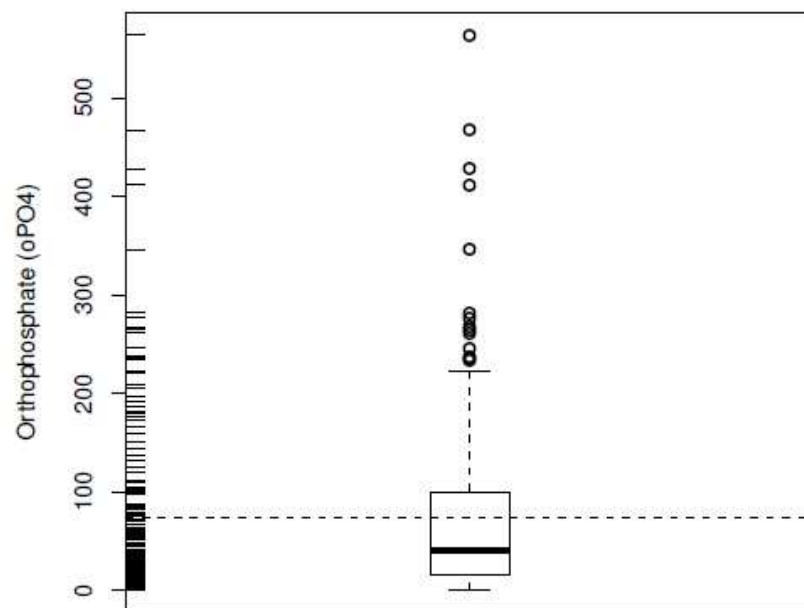
```
> library(car)
> par(mfrow=c(1,2))
> hist(algae$mxPH, prob=T, xlab='',
+      main='Histogram of maximum pH value',ylim=0:1)
> lines(density(algae$mxPH,na.rm=T))
> rug(jitter(algae$mxPH))
> qq.plot(algae$mxPH,main='Normal QQ plot of maximum pH')
> par(mfrow=c(1,1))
```

Después de cargar el paquete, el código comienza llamando a la **función** `par()` que es utilizado en este caso para dividir la ventana de salida del gráfico en dos columnas, con el objetivo de de obtener dos gráficos, uno al lado del otro. El primer gráfico que se obtiene es nuevamente un histograma de la variable **mxPH**, excepto que en este caso hay otros límites para el eje Y. la siguiente instrucción dibuja una suave aproximación sobre el histograma para tratar de representar la distribución de la variable. La siguiente instrucción dibuja los valores reales de la variable ceca del eje X. El segundo gráfico muestra el Q-Q plot obtenido con la **función** `qq.plot()`, el cual grafica los valores de la variable contra los cuantiles teóricos de una distribución normal (línea

negra sólida). La función también muestra un intervalo de confianza del 95% de confianza de la distribución normal (línea punteada). Se puede observar que existen varios valores por debajo de dicho intervalo, rompiendo claramente el supuesto de una distribución normal con un 95% de confianza.

Otro ejemplo para la inspección de los datos, puede realizarse a través de las siguientes instrucciones, en este caso se realizarán para la variable **oPO₄**:

```
> boxplot(algae$oP04, ylab = "Orthophosphate (oP04)")  
> rug(jitter(algae$oP04), side = 2)  
> abline(h = mean(algae$oP04, na.rm = T), lty = 2)
```



La primera instrucción dibuja una caja de la variable **oPO₄**. El gráfico box plot provee un rápido resumen de las propiedades claves de la distribución de la variable. Hay una caja, cuyos límites verticales son el 1er y el 3er cuartil de la variable. Esta caja tiene una línea horizontal dentro, que representa el valor de la mediana de la variable. Sea r el rango inter-cuartílico. La línea horizontal más pequeña por encima de la caja, es la observación más grande, la cual es menor o igual al 3er cuartil más $1,5 \cdot r$. La línea horizontal más pequeña por debajo de la caja, es la observación más pequeña, la cual es más grande o igual que el 1er cuartil más $1,5 \cdot r$. Los círculos debajo o encima de estas pequeñas líneas representan las observaciones que son extremadamente bajas (altas) comparadas con las otras, y son usualmente consideradas como outliers (puntos extraños o

espurios). Esto significa, que el diagrama de cajas da una información plena, referente, no solamente a los valores centrales y a la dispersión de la variable, sino también a valores outliers.

La segunda instrucción fue descrita anteriormente, mientras que la tercera instrucción utiliza la **función abline()** para dibujar una línea horizontal como el valor medio de la variable, el cual es obtenido utilizando la **función mean()**. Comparando esta línea con la línea dentro de la caja que denota la mediana, se puede concluir que la presencia de varios puntos outliers ha distorsionado el valor de la media como un estadístico de tendencia central.

El análisis de la figura anterior, muestra que la variable **oPO₄** tiene una distribución de los valores observados claramente concentrado sobre los valores bajos, esto con una asimetría positiva. En la mayoría de las muestras de agua, el valor de **oPO₄** es bajo, pero hay varias observaciones con valores altos, y otros valores aun extremadamente más altos.

Algunas veces cuando encontramos puntos outliers, es de interés inspeccionar las observaciones que tienen ese comportamiento “extraño”. A continuación se mostrará la manera gráfica de realizarlo. Si se dibujan los valores de la variable **NH₄**, se puede observar un valor muy grande. Se puede identificar la respectiva muestra de agua a través de las siguientes instrucciones:

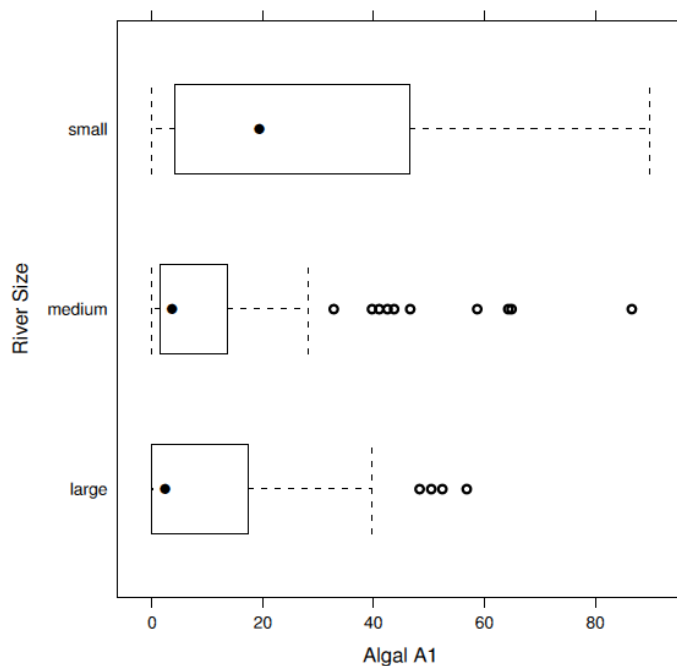
```
> plot(algae$NH4, xlab = "")
> abline(h = mean(algae$NH4, na.rm = T), lty = 1)
> abline(h = mean(algae$NH4, na.rm = T) + sd(algae$NH4, na.rm = T),
+       lty = 2)
> abline(h = median(algae$NH4, na.rm = T), lty = 3)
> identify(algae$NH4)
```

La primera instrucción grafica todos los valores de la variable. Las llamadas a la **función abline()** dibuja tres líneas informativas, una con el valor medio, una con la media mas una desviación estándar y la otra con la mediana. La última instrucción es interactiva y permite al usuario hacer “click” sobre los puntos de la gráfica con el botón izquierdo del mouse. Por cada “click” que se ejecuta sobre un punto, R escribirá el respectivo número de fila en el data frame **algas**. El usuario puede finalizar la interacción haciendo “click” con el botón derecho del mouse.

A continuación se explorarán unos ejemplos de otro tipo de inspección de datos. Estos ejemplos utilizarán el paquete de R **lattice**, el cual provee un gran conjunto de herramientas gráficas impresionantes.

Suponga que se desea estudiar la distribución de los valores del tipo de alga **a1**. Se podrían utilizar cualquiera de las opciones presentadas anteriormente, sin embargo, ya que se desea estudiar como esta distribución depende de otras variables se requieren de nuevas herramientas.

Los gráficos condicionados son representaciones gráficas que dependen de un cierto factor. Un factor es una variable nominal con un conjunto de valores finitos. Por ejemplo, se puede obtener un conjunto de gráficos de caja para la variable **a1**, para cada valor de la variable **tamaño del río**.



Cada uno de los gráficos de caja se obtuvo utilizando un subconjunto de muestras de agua que tienen un cierto valor de la variable **tamaño del río**. Estos gráficos permiten estudiar como la variable nominal puede influenciar la distribución de los valores de la variable **a1**.

El código para generar los diagramas de caja mostrados anteriormente se muestra a continuación:

```
> library(lattice)
> bwplot(size ~ a1, data=algae, ylab='River Size',xlab='Algal A1')
```

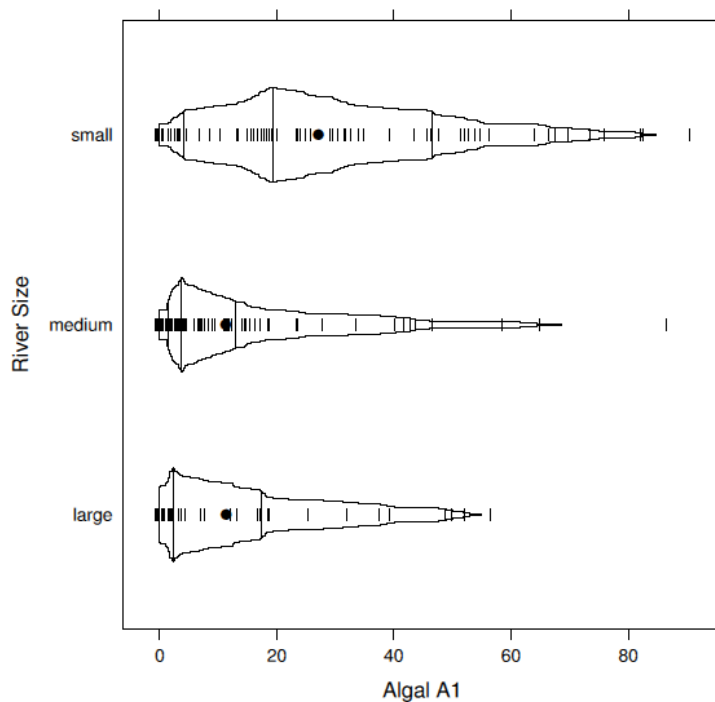
La primera instrucción carga el paquete **lattice**, la segunda genera los diagramas de caja utilizando la versión de **lattice** para estos gráficos.

La figura anterior, permite observar que la mayor frecuencia de ocurrencia del tipo de alga **a1** se espera que ocurra en los ríos de menor tamaño, lo cual aporta valioso conocimiento al estudio.

Una variante interesante de este tipo de gráficos, que puede dar aun más información acerca de la distribución de la variable, son los gráficos de caja-percentiles, los cuales están disponibles en el paquete **Hmisc**. A continuación se presentará un ejemplo con el mismo tipo de alga **a1** contra el **tamaño del río**.

```
> library(Hmisc)
> bwplot(size ~ a1, data=algae,panel=panel.bpplot,
+         probs=seq(.01,.49,by=.01), datadensity=TRUE,
+         ylab='River Size',xlab='Algal A1')
```

El gráfico resultante se muestra a continuación.



Los puntos representan el valor medio de la frecuencia del tipo de alga para los diferentes tipos de tamaño del río. Las líneas verticales representan el 1er cuartil, la mediana y el 3er cuartil, respectivamente. El gráfico muestra los valores de la variable con pequeñas rayas, y la información de la distribución de estos valores es proporcionado por el gráfico de los cuantiles. Estos gráficos proporcionan más información que el gráfico de cajas estándar mostrados anteriormente. Por ejemplo, se puede confirmar la conclusión previa de que los ríos con tamaño más pequeño tienen más alta frecuencia de ocurrencia de este tipo de alga, y adicionalmente se puede observar que los valores de las observaciones para estos ríos pequeños, están mucho más dispersos a través del dominio de las frecuencias que los otros tipos de ríos.

Este tipo de gráficos condicionados no está restringido solamente a variables nominales, ni de un solo factor. Se pueden llevar a cabo este mismo tipo de estudio condicionados con variables continuas siempre y cuando se discreticen previamente.

2.5. Valores desconocidos.

Hay varias muestras de agua con valores desconocidos en alguna de sus variables. Esta situación es muy común en muchos problemas del mundo real, y puede evitar el uso de ciertas técnicas que no están disponibles para manejar valores faltantes o perdidos.

Existen varias técnicas para manejar el caso de valores faltantes, las más comunes se mencionan a continuación:

- Remover los casos con valores faltantes.
- Llenar el valor faltante explorando las correlaciones entre variables.
- Llenar el valor faltante explorando la similaridad entre casos.
- Usar herramientas disponibles para manejar estos valores.

En las siguientes subsecciones se mostrarán ejemplos de como implementar estas estrategias en R. Si desea ejecutar el código en estas secciones debe recordar que no son complementarias, es decir, que antes de trabajar con alguna otra técnica de valores faltantes debe leer nuevamente los datos originales que contengan todos los valores desconocidos, ya que cada sección maneja esta situación de manera diferente. La forma más fácil de realizarlo (cargar nuevamente los datos originales) es ejecutar el siguiente código:

```
> library(DMwR)
> data(algae)
```

2.5.1. Removiendo las observaciones con valores faltantes o desconocidos.

La opción de remover los casos con valores desconocidos es muy fácil de implementar, y también una opción razonable cuando la proporción de los casos con datos faltantes es pequeña respecto al tamaño del conjunto de datos disponible.

Antes de eliminar todas las observaciones con al menos un valor faltante en alguna variable, es recomendable conocer cuantas observaciones presentan al menos un valor faltante, para lo que se debe ejecutar el siguiente código:

```
> algae[!complete.cases(algae),]
...
...
> nrow(algae[!complete.cases(algae),])
[1] 16
```

Donde la instrucción anterior presenta cuantas muestras de agua con algún valor faltante (NA) existen.

La orden para remover las 16 muestras de agua del data frame es la siguiente:

```
> algae <- na.omit(algae)
```

2.5.2. Llenar los valores faltantes con los valores más frecuentes.

Una alternativa para eliminar los casos con valores faltantes es tratar de encontrar el valor más probable para cada uno de esos valores desconocidos.

La manera más simple y rápida de llenar unos valores desconocidos es utilizar algún estadístico de tendencia central. Estos estadísticos reflejan el valor más frecuente de la distribución de una variable. Existen varios estadísticos de tendencia central, como la media, la mediana, la moda, entre otros. La selección del valor más adecuado depende de la distribución de la variable. Para distribuciones aproximadamente normal, donde todas las observaciones están agrupadas alrededor de la media, este estadístico (**la media**) es la mejor opción. Sin embargo, para distribuciones asimétricas que tiene la mayoría de los datos agrupados para uno de los lados del rango de valores de la variable, la media, claramente, no es representativa de los valores más comunes. Por otra parte, la presencia de datos outliers (valores extremos) pueden distorsionar el cálculo de la media, presentándose problemas similares con la falta de representatividad. Para distribuciones asimétricas o para variables con datos outliers, **la mediana** es el mejor estadístico de tendencia central.

Por ejemplo, la muestra algae[48,] no tiene un valor en la variable **mxPH**. Como la distribución de esta variable es claramente normal se puede utilizar el valor de la media para llenar este valor faltante. Lo anterior se realiza a través de la siguiente instrucción:

```
> algae[48, "mxPH"] <- mean(algae$mxPH, na.rm = T)
```

La mayoría de las veces es de interés llenar todos los valores desconocidos de una columna, en lugar de trabajar caso por caso, como se hizo en el ejemplo anterior. Veamos un ejemplo, de esto con la variable **Chla**. Esta variable tiene valores desconocidos para 12 muestras de agua. Además, es una situación, en la que la media representa pobremente los valores más frecuentes de la variable. En efecto, la distribución de la variable **Chla** es asimétrica, y presenta algunos valores extremos que hacen que el valor de la media no sea representativa de los valores más frecuentes. Por lo tanto, se utilizará la mediana para sustituir los valores desconocidos para esta columna, a través del siguiente comando:

```
> algae[is.na(algae$Chla), "Chla"] <- median(algae$Chla, na.rm = T)
```

La función **centralImputation()**, sustituye todos los valores faltantes en un conjunto de datos utilizando un estadístico de tendencia central. Esta función utiliza la mediana para las columnas numéricas y utiliza el valor más frecuente (la moda) para variables nominales. Con los siguientes comandos que se muestran a continuación:

```
> data(algae)
> algae <- algae[-manyNAs(algae), ]
> algae <- centralImputation(algae)
```

Mientras la presencia de valores desconocidos puede impedir el uso de algunos métodos, sustituir estos valores utilizando una técnica como las anteriores se considera mala idea. Esta simple estrategia, además de rápida, y atractiva para un conjunto grande de datos, puede introducir error en los datos, lo cual puede influir en análisis posteriores. Sin embargo, los métodos de estimación que encuentran los valores óptimos para rellenar esos valores faltantes, son extremadamente complejos y pueden no ser adecuados para algunos problemas grandes de minería de datos.

2.5.3. Llenar un valor desconocido por la exploración de la correlación.

Una alternativa para obtener un estimador de menor varianza de un valor desconocido es explorar la relación entre variables. Por ejemplo, utilizando la correlación entre los valores de las variables se podría descubrir que una cierta variable está altamente correlacionada con **mxPH**, con la cual se podría obtener un valor más probable para el valor faltante de la muestra número 48.

Para obtener la correlación entre las variables se utiliza el siguiente comando:

```
> cor(algae[, 4:18], use = "complete.obs")
```

La **función cor()** produce una matriz con los valores de correlación entre las variables.

Valores cercanos a 1 (-1) indica una fuerte correlación lineal positiva (negativa) entre los valores de las dos respectivas variables. El resultado de la función **cor()** no es muy legible pero se puede mostrar haciendo uso de la **función symnum()** para mejorar este inconveniente. El comando se muestra a continuación:

```

> symnum(cor(algae[,4:18],use="complete.obs"))

      mP m0 C1 NO NH o P Ch a1 a2 a3 a4 a5 a6 a7
mxPH 1
mn02  1
C1    1
NO3    1
NH4    , 1
oP04   . . 1
P04    . . * 1
Chla   . 1
a1     . . . 1
a2     . . 1
a3     . 1
a4     . 1
a5     . 1
a6     . . 1
a7     . 1
attr(,"legend")
[1] 0 ' ' 0.3 '!' 0.6 ', ' 0.8 '+ ' 0.9 '* ' 0.95 'B' 1

```

Esta representación simbólica de los valores de la correlación es más legible, particularmente para matrices de correlaciones grandes.

Para estos datos, las correlaciones son irrelevantes en la mayoría de los casos. Sin embargo, hay dos excepciones: entre las variables NH4 y NO3, y entre PO4 y oPO4. Estas dos últimas variables están fuertemente correlacionadas (por encima de 0,9). La correlación entre NH4 y NO3 es menos evidente (0,72) así que es un riesgo tomar ventaja de esto para rellenar los valores faltantes. Sin embargo, asumiendo que se han removido las muestras 62 y 199 porque tenían muchos valores faltantes, no hay muestras de agua con valores desconocidos para las variables NH4 y NO3. Respecto a PO4 y oPO4, el descubrimiento de esta correlación nos lleva a sustituir los valores desconocidos para esas variables. Para alcanzar esto, se necesita encontrar la forma de la correlación lineal entre estas variables. Esto se puede hacer de la siguiente forma:

```

> data(algae)
> algae <- algae[-manyNAs(algae), ]
> lm(PO4 ~ oPO4, data = algae)

Call:
lm(formula = PO4 ~ oPO4, data = algae)

Coefficients:
(Intercept)          oPO4
    42.897         1.293

```

La **función lm()** puede ser usada para obtener modelos lineales de la forma $Y = \beta_0 + \beta_1 X_1 + \dots + \beta_n X_n$. El cual se describirá esta función con detalle en la sección 2.6. El modelo lineal que se obtuvo es de la forma $PO4 = 42,897 + 1,293 * oPO4$. Con esta fórmula se pueden sustituir los valores faltantes de estas variables, siempre y cuando ambos no sean desconocidos para la misma muestra.

Después de remover la muestra 62 y 199, solo queda una observación con un valor desconocido para la variable PO4 (muestra 28), simplemente se utiliza la relación descubierta para hacer lo siguiente:

```

> algae[28, "PO4"] <- 42.897 + 1.293 * algae[28, "oPO4"]

```

Sin embargo, para propósitos ilustrativos, asumamos que hay varias muestras con valores desconocidos para la variable PO4. Lo mejor que se podría hacer es crear una función que devuelva el valor de PO4 dado el valor de oPO4, y entonces aplicar esta función para todos los valores faltantes, tal como se muestra a continuación:

```

> data(algae)
> algae <- algae[-manyNAs(algae), ]
> fillPO4 <- function(oP) {
+   if (is.na(oP))
+     return(NA)
+   else return(42.897 + 1.293 * oP)
+ }
> algae[is.na(algae$PO4), "PO4"] <- sapply(algae[is.na(algae$PO4),
+   "oPO4"], fillPO4)

```

2.5.4. Llenar los valores desconocidos explorando la similaridad entre los casos.

El enfoque descrito en esta sección asume que si dos muestras de agua son similares, y una de ellas tiene un valor desconocido en alguna variable, hay una alta probabilidad de que ese valor sea similar al valor de la otra muestra. Para utilizar este método intuitivamente, se necesita definir la noción de similaridad. Esta noción es usualmente definida utilizando una medida sobre el espacio multivariante de las variables utilizadas para describir las observaciones. Muchas medidas existen

en la literatura, pero una escogencia común es la distancia Euclideana. La cual puede ser informalmente definida como la raíz de la suma de las diferencias al cuadrado entre los valores de dos casos cualesquiera, es decir:

$$d(x, y) = \sqrt{\sum_{i=1}^p (x_i - y_i)^2}$$

El método descrito a continuación utilizará esta medida para encontrar los 10 casos más similares de cualquier muestra de agua con algún valor faltante en una variable, y después usará esos valores para sustituirlo por el desconocido. Se consideran dos maneras de calcular esos valores. La primera forma, simplemente calcula la mediana de los valores de los 10 vecinos más cercanos para llenar en la brecha. En el caso de variables nominales desconocidas (el cual no ocurre en este conjunto de datos), se usa el valor más frecuente (la moda) entre los vecinos. El segundo método utiliza los pesos promedios de los valores de los vecinos. Los pesos se decrementan cuando la distancia entre los casos vecinos se incrementa. Se utiliza una función Gaussiana kernel para obtener los pesos de las distancias.

Esta idea es implementada en la función **knnImputation()**. La función utiliza una variante de la distancia Euclideana para encontrar los k vecinos más cercanos de cualquier caso.

```
> algae <- knnImputation(algae, k = 10)
```

En el caso de que se prefiera utilizar la estrategia de usar los valores medianos para llenar los faltantes, se podría ejecutar la siguiente instrucción:

```
> algae <- knnImputation(algae, k = 10, meth = "median")
```

2.6. Obteniendo Modelos de Predicción.

La meta principal de este caso de estudio es obtener las predicciones para los valores de la frecuencia de aparición de siete tipo de algas en un conjunto de 140 muestras de agua. Dado que estas frecuencias son numéricas, nos enfrentamos a una tarea de regresión. En palabras sencillas, la tarea consiste en tratar de obtener un modelo que relacione una variable numérica con un conjunto de otras variables explicativas. El modelo puede ser utilizado para predecir los valores de la variable objetivo para futuras observaciones de las variables explicativas, o proveer un mejor entendimiento de las interacciones entre las variables de nuestro problema.

En esta sección se utilizaran dos modelos predictivos, el modelo de regresión lineal y los árboles de regresión, los cuales son buenas alternativas para problemas de regresión, ya que son fáciles de interpretar y corren en cualquier computador. Esto no significa que en el escenario real de minería de dato no se intenten otras alternativas.

2.6.1. Regresión lineal múltiple.

La regresión lineal múltiple está entre las técnicas estadísticas más utilizadas para el análisis de datos. Estos modelos obtienen una función aditiva que relaciona una variable predictiva con un conjunto de variables predictoras. Esta función aditiva es la suma de términos de la forma $\beta_i * X_i$, donde X_i es una variable predictora y β_i es un número.

Antes de aplicar este método, se eliminarán las muestras número 62 y 199, ya que como se mencionó anteriormente tiene seis valores faltantes en las variables predictoras. El siguiente código origina un data frame sin dichos valores faltantes:

```
> data(algae)
> algae <- algae[-manyNAs(algae), ]
> clean.algae <- knnImputation(algae, k = 10)
```

Para obtener un modelo de regresión lineal para predecir la frecuencia de una de las algas se debe ejecutar la siguiente instrucción:

```
> lm.a1 <- lm(a1 ~ ., data = clean.algae[, 1:12])
```

La **función lm()** genera un modelo de regresión lineal. El primer argumento de la función indica la forma funcional del modelo. En este caso se desea un modelo que prediga la variable **a1** utilizando todas las demás variables presentes en la data, lo cual se expresa con el uso del carácter '.' (punto). Por ejemplo, si se desea un modelo para predecir **a1** como una función de las variables **mxPH** y **NH4**, se debería indicar el modelo como "a1-----mxPH + NH4".

El resultado de la función es un objeto que contiene la información del modelo lineal.

Se puede obtener mayor detalle del modelo lineal con la siguiente instrucción:

```

> summary(lm.a1)

Call:
lm(formula = a1 ~ ., data = clean.algae[, 1:12])

Residuals:
    Min       1Q   Median       3Q      Max
-37.679 -11.893  -2.567   7.410  62.190

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  42.942055   24.010879   1.788  0.07537 .
seasonspring   3.726978    4.137741   0.901  0.36892
seasonsummer   0.747597    4.020711   0.186  0.85270
seasonwinter   3.692955    3.865391   0.955  0.34065
sizemedium     3.263728    3.802051   0.858  0.39179
sizeshall      9.682140    4.179971   2.316  0.02166 *
speedlow       3.922084    4.706315   0.833  0.40573
speedmedium    0.246764    3.241874   0.076  0.93941
mxPH           -3.589118    2.703528  -1.328  0.18598
mnO2            1.052636    0.705018   1.493  0.13715
Cl              -0.040172    0.033661  -1.193  0.23426
NO3            -1.511235    0.551339  -2.741  0.00674 **
NH4              0.001634    0.001003   1.628  0.10516
oPO4           -0.005435    0.039884  -0.136  0.89177
PO4            -0.052241    0.030755  -1.699  0.09109 .
Chla           -0.088022    0.079998  -1.100  0.27265
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 17.65 on 182 degrees of freedom
Multiple R-squared:  0.3731,    Adjusted R-squared:  0.3215
F-statistic: 7.223 on 15 and 182 DF,  p-value: 2.444e-12

```

La aplicación de la función `summary()` sobre el modelo lineal da la información del diagnóstico concerniente al modelo obtenido. Al principio, se obtiene información referente a los residuos (los errores) del ajuste del modelo lineal a los datos utilizados. Estos residuos deberían tener media cero y seguir una distribución normal (obviamente ser lo más pequeño posible).

Para cada coeficiente (variable) de la ecuación de regresión múltiple, R mostrará su valor y también su error estándar (una estimación de la variabilidad de estos coeficientes). Para verificar la importancia de cada coeficiente, se puede probar la hipótesis de que cada uno de ellos es nulo, esto es, $H_0: \beta_i = 0$. Para probar esta hipótesis, la prueba t es normalmente utilizada. R muestra una columna ($\text{Pr}(>|t|)$) donde el valor está asociado a cada coeficiente con el nivel al cual la hipótesis nula es rechazada. Así, un valor de 0.0001 significa que con una confianza del 99.99% el coeficiente es no nulo. R marca cada prueba con un símbolo correspondiente a un conjunto de

niveles de confianza comunes utilizados para estas pruebas. En resumen, solo para los coeficientes que tengan algún símbolo en frente de ellos, se puede rechazar la hipótesis de que son nulos con al menos un 90% de confianza.

Otro resultado importante en el coeficiente R^2 (múltiple y ajustado). Este indica el grado de ajuste del modelo a los datos, es decir, la proporción de variabilidad en los datos que es explicada por el modelo. Valores cercanos a 1 es mejor (explica casi el 100% de la variabilidad), mientras que valores más pequeños evidencian una falta de ajuste. El coeficiente ajustado es más exigente ya que toma en cuenta el número de parámetros del modelo de regresión.

Finalmente, también se puede probar la hipótesis nula de que no hay dependencia entre la variable objetivo y alguna de las variables explicativas, esto es, $H_0: \beta_1 = \beta_2 = \dots = \beta_m = 0$. El estadístico F puede ser usado para este propósito comparando con el valor crítico. R proporciona el nivel de confianza con el cual se puede estar seguro que se rechaza la hipótesis nula. Así, un p-value de 0.0001 significa que con un 99.99% de confianza la hipótesis nula no es cierta. Usualmente, si el modelo falla esta prueba, se dice que no tiene sentido revisar la prueba t para cada uno de los coeficientes.

En este caso, la proporción de la variabilidad explicada por el modelo no es muy grande (alrededor del 32%). Aun así, se puede rechazar la hipótesis de que la variable objetivo no depende de los predictores (el p-value de la prueba F es muy pequeño). Revisando la significancia de algunos de los coeficientes, es cuestionable la inclusión de alguno de ellos en el modelo. Existen varios métodos para simplificar los modelos de regresión. En esta sección se explorará un método usualmente conocido como eliminación backward.

Se comenzará el estudio de simplificación del modelo utilizando la **función anova()**. Cuando se aplica dicha función a un modelo lineal simple, esta arroja un análisis de varianza secuencial del ajuste del modelo. Esto es, las reducciones en la suma de cuadrados de los residuos (el error total del modelo) cuando cada término de la fórmula es añadido de vuelta. El resultado de este análisis para el modelo obtenido anteriormente se muestra a continuación.

```
> anova(lm.a1)
```

Analysis of Variance Table

Response: a1

	Df	Sum Sq	Mean Sq	F value	Pr(>F)	
season	3	85	28.2	0.0905	0.9651944	
size	2	11401	5700.7	18.3088	5.69e-08	***
speed	2	3934	1967.2	6.3179	0.0022244	**
mxPH	1	1329	1328.8	4.2677	0.0402613	*
mnO2	1	2287	2286.8	7.3444	0.0073705	**
Cl	1	4304	4304.3	13.8239	0.0002671	***
NO3	1	3418	3418.5	10.9789	0.0011118	**
NH4	1	404	403.6	1.2963	0.2563847	
oP04	1	4788	4788.0	15.3774	0.0001246	***
P04	1	1406	1405.6	4.5142	0.0349635	*
Chla	1	377	377.0	1.2107	0.2726544	
Residuals	182	56668	311.4			

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Este resultado indica que la variable estación (season) es la variable que menos contribuye a la reducción del error ajustado del modelo. A continuación se mostrará como se remueve esta variable del modelo:

```
> lm2.a1 <- update(lm.a1, . ~ . - season)
```

La **función update()** puede ser utilizada para realizar pequeños cambios a un modelo lineal existente. En este caso, se puede obtener un nuevo modelo removiendo la variable estación (season) del modelo lm.a1 obtenido previamente. La información para este nuevo modelo se muestra a continuación:

```
> summary(lm2.a1)
```

Call:
lm(formula = a1 ~ size + speed + mxPH + mnO2 + Cl + NO3 + NH4 +
oPO4 + PO4 + Chla, data = clean.algae[, 1:12])

Residuals:

	Min	1Q	Median	3Q	Max
	-36.460	-11.953	-3.044	7.444	63.730

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	44.9532874	23.2378377	1.934	0.05458 .
sizemedium	3.3092102	3.7825221	0.875	0.38278
sizesmall	10.2730961	4.1223163	2.492	0.01358 *
speedlow	3.0546270	4.6108069	0.662	0.50848
speedmedium	-0.2976867	3.1818585	-0.094	0.92556
mxPH	-3.2684281	2.6576592	-1.230	0.22033
mnO2	0.8011759	0.6589644	1.216	0.22561
Cl	-0.0381881	0.0333791	-1.144	0.25407
NO3	-1.5334300	0.5476550	-2.800	0.00565 **
NH4	0.0015777	0.0009951	1.586	0.11456
oPO4	-0.0062392	0.0395086	-0.158	0.87469
PO4	-0.0509543	0.0305189	-1.670	0.09669 .
Chla	-0.0841371	0.0794459	-1.059	0.29096

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 17.57 on 185 degrees of freedom
Multiple R-squared: 0.3682, Adjusted R-squared: 0.3272
F-statistic: 8.984 on 12 and 185 DF, p-value: 1.762e-13

El ajuste ha mejorado un poco (32.8%) pero aun no es impresionante. Se puede efectuar una comparación formal entre los dos modelos utilizando nuevamente la función `anova()`, pero esta vez ambos modelos se usan como argumento:

```
> anova(lm.a1,lm2.a1)
```

Analysis of Variance Table

Model 1: a1 ~ season + size + speed + mxPH + mnO2 + Cl + NO3 + NH4 +
oPO4 + PO4 + Chla
Model 2: a1 ~ size + speed + mxPH + mnO2 + Cl + NO3 + NH4 + oPO4 +
PO4 + Chla

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
1	182	56668				
2	185	57116	-3	-448	0.4792	0.6971

Esta función realiza un análisis de varianza de los dos modelos utilizando una prueba F para evaluar la significancia de las diferencias. En este caso, la suma cuadrado del error ha disminuido (-448), la comparación muestra que las diferencias son no significativas (un valor de 0.6971 dice que con solo aproximadamente un nivel de confianza del 30% son diferentes). Aunque no existan diferencias entre los modelos este último es más simple. Para verificar si se pueden eliminar más coeficientes, se usaría nuevamente la **función anova()** sobre el modelo lm2.a1. Este proceso se continua hasta que no hayan coeficientes candidatos para eliminar. Sin embargo, para simplificar el proceso de eliminación, R tiene una función que realiza el proceso por si solo.

El siguiente código crea un modelo lineal que resulta de aplicar el método de eliminación backward al modelo inicial que se había obtenido (lm.a1):

```
> final.lm <- step(lm.a1)
```

```
Start: AIC= 1151.85
```

```
a1 ~ season + size + speed + mxPH + mn02 + Cl + N03 + NH4 + oP04 +  
P04 + Chla
```

	Df	Sum of Sq	RSS	AIC
- season	3	425	57043	1147
- speed	2	270	56887	1149
- oP04	1	5	56623	1150
- Chla	1	401	57018	1151
- Cl	1	498	57115	1152
- mxPH	1	542	57159	1152
<none>			56617	1152
- mn02	1	650	57267	1152
- NH4	1	799	57417	1153
- P04	1	899	57516	1153
- size	2	1871	58488	1154
- N03	1	2286	58903	1158

```
Step: AIC= 1147.33
```

```
a1 ~ size + speed + mxPH + mn02 + Cl + N03 + NH4 + oP04 + P04 +  
Chla
```

	Df	Sum of Sq	RSS	AIC
- speed	2	213	57256	1144
- oP04	1	8	57050	1145
- Chla	1	378	57421	1147
- mn02	1	427	57470	1147
- mxPH	1	457	57500	1147
- Cl	1	464	57506	1147
<none>			57043	1147
- NH4	1	751	57794	1148
- P04	1	859	57902	1148
- size	2	2184	59227	1151
- N03	1	2353	59396	1153

...
...

Step: AIC= 1140.09
a1 ~ size + mxPH + Cl + NO3 + PO4

	Df	Sum of Sq	RSS	AIC
<none>			58432	1140
- mxPH	1	801	59233	1141
- Cl	1	906	59338	1141
- NO3	1	1974	60405	1145
- size	2	2652	61084	1145
- PO4	1	8514	66946	1165

La **función step()** utiliza el Criterio de Información Akaike para realizar la búsqueda del modelo. La búsqueda utiliza la eliminación backward por defecto. Se puede obtener la información sobre el modelo final por el siguiente código:

```
> summary(final.lm)
```

Call:

```
lm(formula = a1 ~ size + mxPH + Cl + NO3 + PO4, data = clean.algae[,  
  1:12])
```

Residuals:

Min	1Q	Median	3Q	Max
-28.874	-12.732	-3.741	8.424	62.926

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	57.28555	20.96132	2.733	0.00687	**
sizemedium	2.80050	3.40190	0.823	0.41141	
sizesmall	10.40636	3.82243	2.722	0.00708	**
mxPH	-3.97076	2.48204	-1.600	0.11130	
Cl	-0.05227	0.03165	-1.651	0.10028	
NO3	-0.89529	0.35148	-2.547	0.01165	*
PO4	-0.05911	0.01117	-5.291	3.32e-07	***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 17.5 on 191 degrees of freedom

Multiple R-squared: 0.3527, Adjusted R-squared: 0.3324

F-statistic: 17.35 on 6 and 191 DF, p-value: 5.554e-16

La proporción de variabilidad explicada por el modelo no es aun muy interesante. El tipo de proporción es usualmente considerado una señal de que la suposición de linealidad de este modelo es inadecuada para el dominio.

2.6.2. Árboles de regresión.

A continuación se presenta otro tipo de modelo de regresión disponible en R. Este modelo maneja un conjunto de datos con valores faltantes, solamente se eliminarán las muestras 62 y 199 por las razones mencionadas anteriormente. Las instrucciones necesarias para obtener el árbol de regresión se presentan a continuación:

```
> library(rpart)
> data(algae)
> algae <- algae[-manyNAs(algae), ]
> rt.a1 <- rpart(a1 ~ ., data = algae[, 1:12])
```

La primera instrucción carga el paquete **rpart** implementa el árbol de regresión en R. La última instrucción obtiene el árbol. Note que esta función utiliza el mismo esquema que la función **lm()** para describir la forma funcional del modelo. El segundo argumento de **rpart()** indica la data que se usó para la obtención del modelo.

El contenido del objeto **rt.a1** se muestra a continuación:

```
> rt.a1

n= 198

node), split, n, deviance, yval
* denotes terminal node

1) root 198 90401.290 16.996460
  2) P04>=43.818 147 31279.120 8.979592
    4) C1>=7.8065 140 21622.830 7.492857
      8) oP04>=51.118 84 3441.149 3.846429 *
      9) oP04< 51.118 56 15389.430 12.962500
        18) mn02>=10.05 24 1248.673 6.716667 *
        19) mn02< 10.05 32 12502.320 17.646870
          38) N03>=3.1875 9 257.080 7.866667 *
          39) N03< 3.1875 23 11047.500 21.473910
            78) mn02< 8 13 2919.549 13.807690 *
            79) mn02>=8 10 6370.704 31.440000 *
```



```

5) C1< 7.8065 7 3157.769 38.714290 *
3) P04< 43.818 51 22442.760 40.103920
6) mxPH< 7.87 28 11452.770 33.450000
12) mxPH>=7.045 18 5146.169 26.394440 *
13) mxPH< 7.045 10 3797.645 46.150000 *
7) mxPH>=7.87 23 8241.110 48.204350
14) P04>=15.177 12 3047.517 38.183330 *
15) P04< 15.177 11 2673.945 59.136360 *

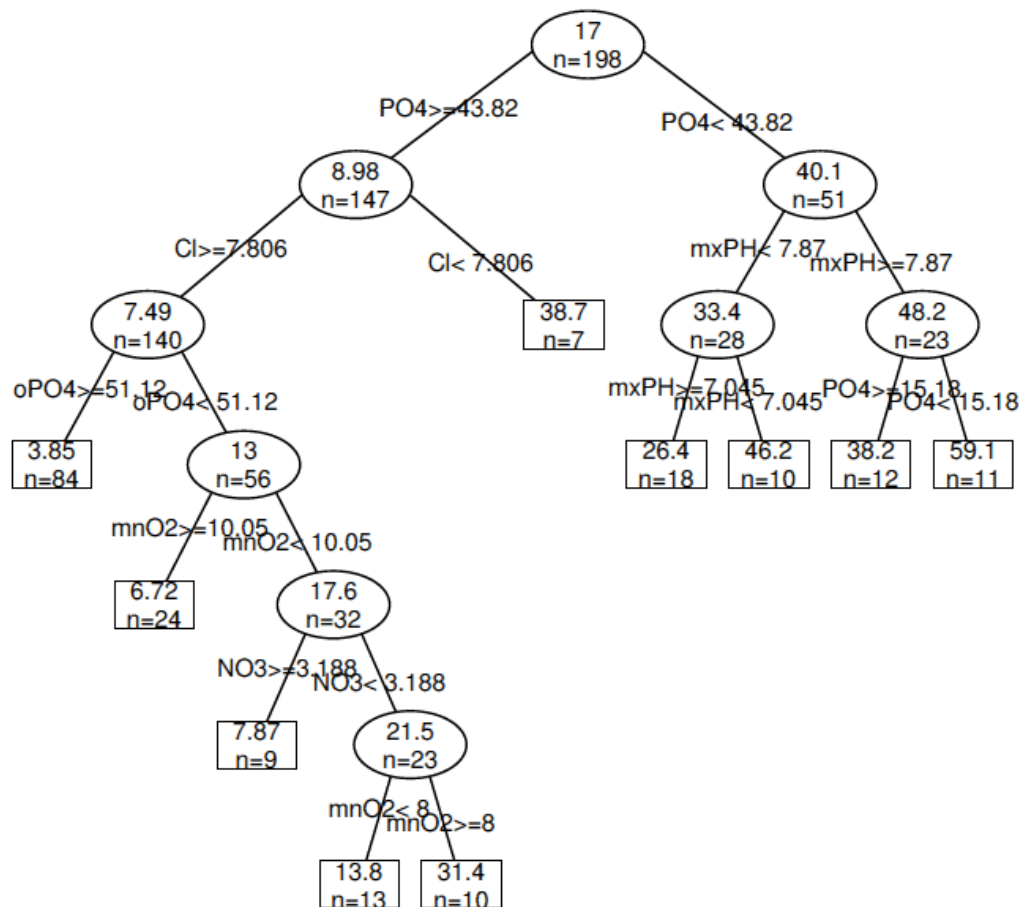
```

Un árbol de regresión es una jerarquía de pruebas lógicas sobre algunas de las variables explicativas. Modelos basados en árbol automáticamente seleccionan las variables más relevantes, y por esta razón, no todas las variables necesitan aparecer en el árbol. El árbol es leído desde el nodo raíz que es marcado por R con el número 1. R provee alguna información de los datos en este nodo. Se puede observar que hay 198 muestras para este nodo (corresponde a los datos de entrenamiento utilizados para generar el árbol), esas 198 muestras tienen un valor promedio para la frecuencia del tipo de alga a1 de 16.99 y su desviación respecto a este valor es 90401.29. Cada nodo del árbol tiene dos ramas, las cuales están relacionadas con el resultado de la prueba sobre una de las variables predictoras. Por ejemplo, desde el nodo raíz se tiene una rama (etiquetada por R con “2”) para los casos donde la prueba “ $P04 \geq 43.818$ ” es verdadera (147 muestras cumplen esa condición) y también una rama para los 51 casos restantes que no satisfacen esa condición (marcada con R con “3”). Desde el nodo 2 se tienen otras dos ramas hacia los nodos 4 y 5, dependiendo del resultado de la prueba sobre la variable C1. Esta prueba continua hasta que un nodo hoja es alcanzado. Estos nodos son marcados con un asterisco por R. estas hojas tienen las predicciones del árbol. Esto significa que si se desea utilizar un árbol para obtener una predicción para una muestra de agua particular, solo se necesita seguir una rama desde el nodo raíz hasta una hoja, de acuerdo al resultado de la prueba para esta muestra. El valor promedio encontrado para la variable objetivo hasta la hoja que se ha alcanzado es la predicción del árbol.

También se puede obtener una representación gráfica del árbol. Esto se puede efectuar aplicando repetidamente las **funciones plot() y text()** para el árbol. Estas funciones tienen varios parámetros para controlar la visualización del árbol. Para facilitar la obtención del gráfico se ha incluido la función **prettyTree()**.

```
> prettyTree(rt.a1)
```

Aplicando esta función se obtiene el árbol, el cual se muestra a continuación:



Existe una estrategia utilizada por R en los arboles de regresión para manejar valores faltantes y es conocida como divisiones o particiones sustitutas.

Los arboles son usualmente obtenidos en dos pasos. Inicialmente, un árbol de gran tamaño está creciendo y después este árbol es podado eliminando los nodos inferiores a través del proceso de estimación estadística. Este proceso tiene como finalidad evitar el sobreajuste. Esto se presenta cuando arboles de gran tamaño se ajustan perfectamente al conjunto de datos de entrenamiento, pero posteriormente presentan un desempeño bajo cuando se enfrentan a nuevos datos para obtener las respectivas predicciones.

La **función rpart()** que se ha utilizado para obtener el árbol, solo hace crecer el árbol, deteniéndose cuando cierto criterio se alcanza. El árbol para de crecer cuando:

- 1) El decremento de la desviación va por debajo de cierto umbral.
- 2) El número de muestras en el nodo es menor que otro umbral.
- 3) La profundidad del árbol excede otro valor.

Estos umbrales son controlados por los parámetros **cp**, **minsplit** y **maxdepth**, respectivamente. Por defecto estos valores son 0.01, 20 y 30, respectivamente. Si se desea evitar el problema del sobreajuste se debe verificar la validez de estos criterios.

El paquete **rpart** implementa un método para podar llamado costo de complejidad de podar. Este método utiliza el valor del parámetro **cp** que R calcula para cada nodo del árbol. Este método para podar trata de estimar el valor de **cp** que asegura el mejor compromiso entre la precisión predictiva y el tamaño del árbol. Dado un árbol obtenido con la función **rpart()**, R puede producir un conjunto de sub-arboles de este árbol y estimar su desempeño predictivo. Esta información puede ser obtenida utilizando la función **printcp()**.

```
> printcp(rt.a1)
```

```
Regression tree:
```

```
rpart(formula = a1 ~ ., data = algae[, 1:12])
```

```
Variables actually used in tree construction:
```

```
[1] C1 mn02 mxPH NO3 oP04 P04
```

```
Root node error: 90401/198 = 456.57
```

```
n= 198
```

	CP	nsplit	rel error	xerror	xstd
1	0.405740	0	1.00000	1.00932	0.12986
2	0.071885	1	0.59426	0.73358	0.11884
3	0.030887	2	0.52237	0.71855	0.11518
4	0.030408	3	0.49149	0.70161	0.11585
5	0.027872	4	0.46108	0.70635	0.11403
6	0.027754	5	0.43321	0.69618	0.11438
7	0.018124	6	0.40545	0.69270	0.11389
8	0.016344	7	0.38733	0.67733	0.10892
9	0.010000	9	0.35464	0.70241	0.11523

El árbol producido por la función **rpart()** es el ultimo árbol de esa lista (árbol 9). Este árbol tiene un valor de **cp** de 0.01 (el valor por defecto de ese parámetro), incluye nueve pruebas y tiene un error relativo (comparado con el nodo raíz) de 0.354. Sin embargo, R estima, utilizando un proceso interno de 10 veces la validación cruzada que este árbol tendrá en promedio un error relativo de 0.70241 ± 0.11523 . Utilizando la información provista por estas estimaciones de desempeño más confiables, la cual evita el problema de sobreajuste, se puede observar que teóricamente podría

tener un mejor ajuste con el árbol número 8, el cual tiene un error relativo estimado menor (0.67733). Una regla de selección alternativa es escoger el mejor árbol de acuerdo a la regla 1-SE, la cual consiste en mirar el error estimado para la validación cruzada (columna "xerror") y su desviación estándar (columna "xstd").

```
> rt2.a1 <- prune(rt.a1, cp = 0.08)
> rt2.a1

n= 198

node), split, n, deviance, yval
  * denotes terminal node

1) root 198 90401.29 16.996460
  2) P04>=43.818 147 31279.12  8.979592 *
  3) P04< 43.818 51 22442.76 40.103920 *
```

La función **rpartXse()** automatiza este proceso y toma como argumento ese valor, por defecto es 1:

```
> (rt.a1 <- rpartXse(a1 ~ ., data = algae[, 1:12]))

n= 198

node), split, n, deviance, yval
  * denotes terminal node

1) root 198 90401.29 16.996460
  2) P04>=43.818 147 31279.12  8.979592 *
  3) P04< 43.818 51 22442.76 40.103920 *
```

R también tiene una especie de poda interactiva a través de la **función snip.rpart()**. Esta función puede ser utilizada para generar la poda del árbol de dos maneras. La primera consiste en indicar el número de nodos (se puede obtener estos números imprimiendo el objeto árbol) en el que se desea podar el árbol:

```
> first.tree <- rpart(a1 ~ ., data = algae[, 1:12])
> snip.rpart(first.tree, c(4, 7))
```

```
n= 198
```

```
node), split, n, deviance, yval
    * denotes terminal node
```

```
1) root 198 90401.290 16.996460
 2) P04>=43.818 147 31279.120 8.979592
   4) C1>=7.8065 140 21622.830 7.492857 *
   5) C1< 7.8065 7 3157.769 38.714290 *
 3) P04< 43.818 51 22442.760 40.103920
   6) mxPH< 7.87 28 11452.770 33.450000
    12) mxPH>=7.045 18 5146.169 26.394440 *
    13) mxPH< 7.045 10 3797.645 46.150000 *
   7) mxPH>=7.87 23 8241.110 48.204350 *
```

Alternativamente, se puede usar **snip.rpart()** de manera gráfica. Primero se dibuja el árbol, y después se llama a la función sin el segundo argumento. Si se hace “click” con el mouse sobre algún nodo, R imprime sobre la consola alguna información acerca del nodo. Si se hace “click” nuevamente sobre el nodo, R poda el árbol en ese nodo. De esta manera se puede ir podando nodos de manera gráfica. Se finaliza la interacción haciendo “click” con el botón derecho del mouse. El resultado de esta llamada es nuevamente un objeto tipo árbol:

```
> prettyTree(first.tree)
> snip.rpart(first.tree)
```

```
node number: 2  n= 147
  response= 8.979592
  Error (dev) = 31279.12
node number: 6  n= 28
  response= 33.45
  Error (dev) = 11452.77
n= 198
```

```
node), split, n, deviance, yval
```

* denotes terminal node

- 1) root 198 90401.290 16.996460
- 2) P04>=43.818 147 31279.120 8.979592 *
- 3) P04< 43.818 51 22442.760 40.103920
- 6) mxPH< 7.87 28 11452.770 33.450000 *
- 7) mxPH>=7.87 23 8241.110 48.204350
- 14) P04>=15.177 12 3047.517 38.183330 *
- 15) P04< 15.177 11 2673.945 59.136360 *

Para el ejemplo, se ha hecho “click” y podado los nodos 2 y 6.

2.7. Evaluación y selección de modelos.

Existen varios criterios para la evaluación y comparación de modelos, siendo el más popular el cálculo del desempeño predictivo. Aun así, existen otros criterios tales como la interpretabilidad del modelo, o la eficiencia computacional, que puede ser importante para problemas grandes de minería de datos.

El desempeño predictivo de los modelos de regresión es obtenido comparando las predicciones de los modelos con los valores reales de las variables objetivo, y calcular alguna medida de error promedio de esta comparación. Tal medida es el error medio absoluto (MAE, por sus siglas en inglés). Para los ejemplos que se han venido presentando se tendrá esta medida para los dos modelos obtenidos (regresión lineal y árbol de regresión). El primer caso es obtener las predicciones del modelo para el conjunto de casos donde se quiere evaluar. Para obtener las predicciones de cualquier modelo en R, se utiliza la **función predict()**. La cual recibe un modelo y prueba un conjunto de datos arrojando las correspondientes predicciones del modelo:

```
> lm.predictions.a1 <- predict(final.lm, clean.algae)
> rt.predictions.a1 <- predict(rt.a1, algae)
```

Estas dos instrucciones poseen las predicciones de los modelos obtenidos en la sección 2.6 para el tipo de alga a1. Note que se ha usado la instrucción clean.algae sobre el data frame del modelo lineal debido a los datos faltantes. Una vez que se tienen las predicciones de los modelos, se calcula su error absoluto medio como se muestra a continuación:

```
> (mae.a1.lm <- mean(abs(lm.predictions.a1 - algae[, "a1"])))
[1] 13.10681

> (mae.a1.rt <- mean(abs(rt.predictions.a1 - algae[, "a1"])))
[1] 11.61717
```

Otra conocida medida para el error es el error cuadrático medio (MSE, por sus siglas en inglés). Esta medida se obtiene de la siguiente manera:

```
> (mse.a1.lm <- mean((lm.predictions.a1 - algae[, "a1"])^2))
```

```
[1] 295.5407
```

```
> (mse.a1.rt <- mean((rt.predictions.a1 - algae[, "a1"])^2))
```

```
[1] 271.3226
```

La desventaja de este estadístico es que no está siendo medido en las mismas unidades que la variable objetivo, y así es menos interpretable desde la perspectiva del usuario. Aun si se utiliza el estadístico MAE, la pregunta es ¿Cuándo el valor obtenido es considerado como bueno o malo?. Una alternativa estadística que provee una respuesta razonable es el error cuadrático medio normalizado (NMSE, por sus siglas en inglés). Este estadístico calcula un ratio entre el desempeño de los modelos y una línea base predictora, usualmente tomada como el valor medio de la variable objetivo, el código se muestra a continuación:

```
> (nmse.a1.lm <- mean((lm.predictions.a1-algae[, 'a1'])^2)/  
+ mean((mean(algae[, 'a1'])-algae[, 'a1'])^2))
```

```
[1] 0.6473034
```

```
> (nmse.a1.rt <- mean((rt.predictions.a1-algae[, 'a1'])^2)/  
+ mean((mean(algae[, 'a1'])-algae[, 'a1'])^2))
```

```
[1] 0.5942601
```

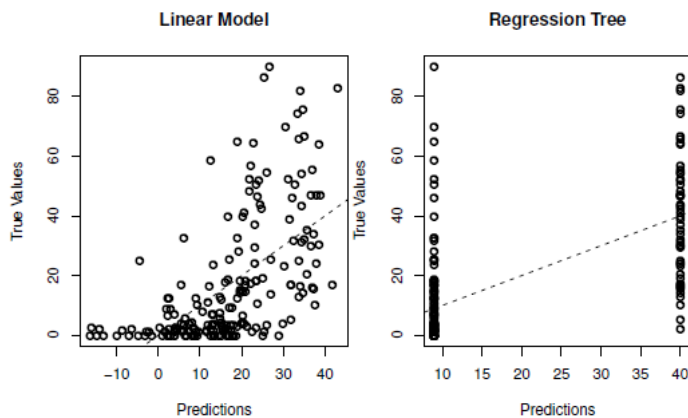
El NMSE es una medida de error adimensional (sin unidades), con valores usualmente entre 0 y 1. Si el modelo se desempeña mejor que su línea de base predictora, entonces el valor de NMSE debería ser menor que 1. Mientras más pequeño el valor de NMSE mejor. Valores mayores que 1 significa que el modelo se desempeña peor simplemente predecir siempre el valor promedio para todos los casos.

La **función `regr.eval()`** calcula el valor de un conjunto de evaluaciones métricas de la regresión.

```
> regr.eval(algae[, "a1"], rt.predictions.a1, train.y = algae[,  
+ "a1"])
```

	mae	mse	rmse	nmse	nmae
	11.6171709	271.3226161	16.4718735	0.5942601	0.6953711

Es también de interés tener algún tipo de inspección visual de las predicciones del modelo. Una posibilidad es utilizar una gráfica de los errores.



La figura muestra un ejemplo de este tipo de análisis para las predicciones de los dos modelos, y se produjo con el siguiente código:

```
> old.par <- par(mfrow = c(1, 2))
> plot(lm.predictions.a1, algae[, "a1"], main = "Linear Model",
+      xlab = "Predictions", ylab = "True Values")
> abline(0, 1, lty = 2)
> plot(rt.predictions.a1, algae[, "a1"], main = "Regression Tree",
+      xlab = "Predictions", ylab = "True Values")
> abline(0, 1, lty = 2)
> par(old.par)
```

Observando la figura se puede apreciar que los modelos presentan un pobre desempeño en ambos casos. El escenario ideal, donde se hicieran correctamente las predicciones para todos los casos, se identificaría por la ubicación de todos los círculos sobre la línea punteada, la cual se genera con el comando **abline(0,1,lty=2)**. Dado que cada círculo en el gráfico obtiene su coordenada de los valores predichos y verdaderos de la variable objetivo, si esos valores son iguales, los círculos deberían estar ubicados sobre la línea. Como se puede observar, este no es el caso. Se puede verificar cual es el número de la muestra donde se realizó particularmente una mala predicción, esto se efectúa a través de la **función identify()**, la cual se ha utilizado haciendo “click” interactivamente sobre los puntos del gráfico, el código se muestra a continuación:

```
> plot(lm.predictions.a1, algae[, 'a1'], main = "Linear Model",
+      xlab = "Predictions", ylab = "True Values")
> abline(0, 1, lty = 2)
> algae[identify(lm.predictions.a1, algae[, 'a1']),]
```

Mirando la figura anterior (parte izquierda) con las predicciones del modelo lineal, se puede apreciar que este modelo predice frecuencias negativas para las algas en algunos casos. En este dominio de la aplicación, no tiene sentido decir que una ocurrencia de una alga en una muestra de agua sea negativa (como mínimo, puede ser cero). Así, se puede tomar ventaja y utilizar el mínimo

valor como una forma de mejorar el desempeño del modelo lineal, lo cual se efectúa a través del siguiente código:

```
> sensible.lm.predictions.a1 <- ifelse(lm.predictions.a1 <
+   0, 0, lm.predictions.a1)
> regr.eval(algae[, "a1"], lm.predictions.a1, stats = c("mae",
+   "mse"))

      mae      mse
13.10681 295.54069

> regr.eval(algae[, "a1"], sensible.lm.predictions.a1, stats = c("mae",
+   "mse"))

      mae      mse
12.48276 286.28541
```

Se utilizó la **función ifelse()** para alcanzar este efecto. Esta función tiene tres argumentos. El primero es una condición lógica, el segundo es el resultado de una llamada a la función cuando la condición es verdadera, mientras que el tercer argumento es el resultado cuando la condición es falsa. Note como este pequeño detalle ha incrementado el desempeño del modelo.

De acuerdo a medidas de desempeño calculadas previamente, se debería preferir el árbol de regresión para obtener las predicciones para las 140 muestras de prueba ya que se obtuvo un menor valor de NMSE. Sin embargo, se ha analizado la capacidad predictiva del modelo sobre los mismos datos de entrenamiento, y el factor clave es obtener una estimación confiable del desempeño de un modelo sobre datos para los cuales no se conocen los verdaderos valores de la variable objetivo. La validación cruzada k-fold está entre los métodos frecuentemente utilizados para obtener estas estimaciones confiables para pequeños conjuntos de datos como en este caso de estudio. Este método puede ser descrito brevemente como se describe a continuación. Se obtienen k subconjuntos aleatorios y de igual tamaño, de los datos de entrenamiento. Para cada uno de esos k subconjuntos, se construye un modelo utilizando los k-1 subconjuntos restantes y se valúa este modelo para el subconjunto k. Se almacena el desempeño del modelo y se repite el proceso para los subconjuntos restantes. Al final, se tienen k medidas de desempeño, todas obtenidas probando un modelo sobre datos no utilizados para su construcción, y ese es el factor clave. La validación cruzada k-fold estima el promedio de estas k medidas. Una escogencia común es tomar k igual a 10. Algunas veces se repite el proceso varias veces para obtener estimaciones más confiables.

La **función experimentalComparison()** es designada para ayudar en la tarea de selección/comparación de modelos. Esta puede ser usada con diferentes métodos de estimación, incluyendo validación cruzada. La función tiene tres parámetros: 1) el conjunto de datos que se utiliza para la comparación, 2) los modelos alternativos, y 3) los parámetros del proceso experimental.

El código se muestra a continuación:

```

> cv.rpart <- function(form,train,test,...) {
+   m <- rpartXse(form,train,...)
+   p <- predict(m,test)
+   mse <- mean((p- resp(form,test))^2)
+   c(nmse=mse/mean((mean(resp(form,train))-resp(form,test))^2))
+ }
> cv.lm <- function(form,train,test,...) {
+   m <- lm(form,train,...)
+   p <- predict(m,test)
+   p <- ifelse(p < 0,0,p)
+   mse <- mean((p- resp(form,test))^2)
+   c(nmse=mse/mean((mean(resp(form,train))-resp(form,test))^2))
+ }

```

En este ejemplo ilustrativo, se asume que se desea utilizar el NMSE como evaluación métrica del árbol de regresión y los modelos lineales.

Habiendo definido las funciones que se encargaran del aprendizaje y fase de prueba de los modelos, se llevará a cabo la comparación validación cruzada como sigue:

```

> res <- experimentalComparison(
+   c(dataset(a1 ~ .,clean.algae[,1:12],'a1')),
+   c(variants('cv.lm'),
+     variants('cv.rpart',se=c(0,0.5,1))),
+   cvSettings(3,10,1234))

```

CROSS VALIDATION EXPERIMENTAL COMPARISON

** DATASET :: a1

++ LEARNER :: cv.lm variant -> cv.lm.defaults

Repetition 1

Fold: 1 2 3 4 5 6 7 8 9 10

Repetition 2

Fold: 1 2 3 4 5 6 7 8 9 10

Repetition 3

Fold: 1 2 3 4 5 6 7 8 9 10

++ LEARNER :: cv.rpart variant -> cv.rpart.v1

Repetition 1

Fold: 1 2 3 4 5 6 7 8 9 10

Repetition 2

Fold: 1 2 3 4 5 6 7 8 9 10

Repetition 3

Fold: 1 2 3 4 5 6 7 8 9 10

++ LEARNER :: cv.rpart variant -> cv.rpart.v2

Repetition 1

Fold: 1 2 3 4 5 6 7 8 9 10

Repetition 2

Fold: 1 2 3 4 5 6 7 8 9 10

Repetition 3

Fold: 1 2 3 4 5 6 7 8 9 10

++ LEARNER :: cv.rpart variant -> cv.rpart.v3

Repetition 1

Fold: 1 2 3 4 5 6 7 8 9 10

Repetition 2

Fold: 1 2 3 4 5 6 7 8 9 10

Repetition 3

Fold: 1 2 3 4 5 6 7 8 9 10

La siguiente instrucción provee un resumen de los resultados de la comparación:

```

> summary(res)

== Summary of a Cross Validation Experiment ==

3 x 10 - Fold Cross Validation run with seed = 1234

* Datasets :: a1
* Learners  :: cv.lm.defaults, cv.rpart.v1, cv.rpart.v2, cv.rpart.v3

* Summary of Experiment Results:

-> Dataset: a1

      *Learner: cv.lm.defaults
            nmse
avg      0.7196105
std      0.1833064
min      0.4678248
max      1.2218455
invalid  0.0000000

      *Learner: cv.rpart.v1
            nmse
avg      0.6440843
std      0.2521952
min      0.2146359
max      1.1712674
invalid  0.0000000

      *Learner: cv.rpart.v2
            nmse
avg      0.6873747
std      0.2669942

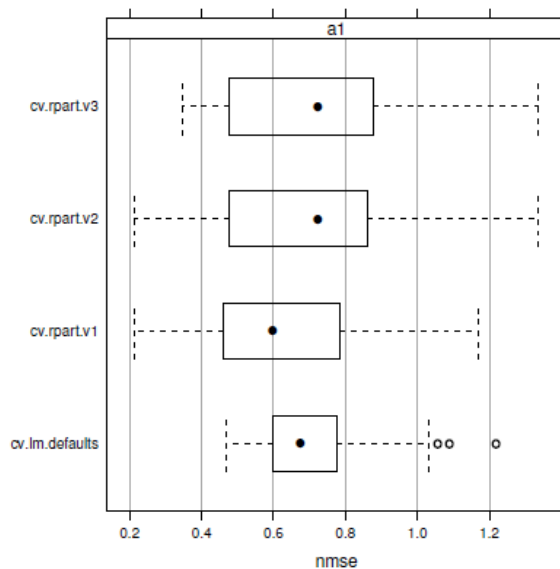
min      0.2146359
max      1.3356744
invalid  0.0000000

      *Learner: cv.rpart.v3
            nmse
avg      0.7167122
std      0.2579089
min      0.3476446
max      1.3356744
invalid  0.0000000

```

Como se puede ver, una de las variantes del árbol de regresión alcanzó el mejor NMSE promedio. Donde si la diferencia es estadísticamente significativa respecto a las otras alternativas, es una pregunta que se tratará posteriormente en este capítulo. También se puede obtener una visualización de estos resultados a través del siguiente comando:

```
> plot(res)
```



2.8. Predicciones para los siete tipos de algas.

En esta sección se mostrará como obtener las predicciones para los siete tipos de algas sobre las 140 muestras de prueba. La sección 2.7 describió como proceder para escoger los mejores modelos para obtener esas predicciones. El procedimiento utilizado consistió en obtener estimadores insesgados del NMSE para un conjunto de modelos sobre las siete predicciones, por medio del proceso experimental de validación cruzada.

La meta principal en este problema de minería de datos es obtener las siete predicciones para cada una de las 140 muestras de prueba. Cada una de estas siete predicciones se obtendrán utilizando el modelo que el proceso de validación cruzada ha indicado como el “mejor” para esta tarea. Será cualquier modelo resultante del llamado a la **función bestScores()** en la sección previa. Será cualquiera de estos: “cv.rf.v3”, “cv.rf.v2”, “cv.rf.v1”, o “cv.rpart.v3”.

Se comenzará obteniendo estos modelos utilizando todos los datos de entrenamiento que se puedan aplicar para el conjunto de prueba. Note que, por simplicidad, crecerá el árbol de regresión utilizando **clean.algae** sobre el data frame que tenía los valores NA sustituidos por el proceso de imputación de los k vecinos más cercanos. Esto puede ser evitado para el árbol de regresión incorporando su propio método para lidiar con valores desconocidos. Los bosques aleatorios, por el contrario, no incluye tal método, y necesitan aprender utilizando el data frame **clean.algae**.

El siguiente código obtiene todos los siete modelos:

```

> bestModelsNames <- sapply(bestScores(res.all),
+                             function(x) x['nmse','system'])
> learners <- c(rf='randomForest',rpart='rpartXse')
> funcs <- learners[sapply(strsplit(bestModelsNames,'\\.'),
+                             function(x) x[2])]
> parSetts <- lapply(bestModelsNames,
+                     function(x) getVariant(x,res.all)@pars)
> bestModels <- list()
> for(a in 1:7) {
+   form <- as.formula(paste(names(clean.algae)[11+a],'~ .'))
+   bestModels[[a]] <- do.call(funcs[a],
+                               c(list(form,clean.algae[,c(1:11,11+a)]),parSetts[[a]]))
+ }

```

Se comienza obteniendo un vector con los nombres de las variantes ganadoras para cada tarea. Después se obtienen los respectivos nombres de las funciones de R que aprenden estas variantes sobre el vector **funcs**. Esto se alcanza extrayendo una parte del nombre de la variable con la **función strsplit()**. Este paso es un ejemplo ligeramente más sofisticado de la función composición, podría ser útil ejecutar este código en partes separadas para entender totalmente el rol de las llamadas a las diferentes llamadas a funciones. La lista **parSetts** es asignada con los parámetros ajustados para cada una de las variantes ganadoras. La función **getVariant()** da el modelo correspondiente al nombre de la variable. El objeto retornado por esta función es de la clase **learner**. Estos objetos tienen diferentes “ranuras”, una de las cuales es nombrada **pars** y contiene una lista con los parámetros de la variante. Las ranuras de un objeto pueden ser obtenidas en R por el operador “@”. Finalmente, se obtienen los modelos y se recogen en la lista **bestModels**. Para cada alga, se construye la fórmula como antes y después se llama con la respectiva función de R utilizando la configuración adecuada de los parámetros. Esto es alcanzado con la **función do.call()** que nos permite llamar a cualquier función, proporcionando su nombre como unos caracteres en el primer argumento, y después incluir los argumentos de la llamada como una lista en el segundo argumento. Después de la ejecución de este código, se tiene una lista con los siete modelos obtenidos para cada tipo de alga y se está listo para hacer las predicciones sobre el conjunto de prueba.

El data **frametest.algae**, contiene las 140 muestras de prueba. Este conjunto de datos también incluye valores desconocidos, así, el primer paso será rellenar estos valores faltantes utilizando la misma metodología usada anteriormente. La primera tentación para llevar a cabo esta tarea sería ejecutar la **función knnImputation()** para el data **frametest.algae**. Esto solucionaría el problema de los datos faltantes pero iría ligeramente en contra de una de las reglas de oro de los modelos predictivos, la cual es, no usar ninguna información del conjunto de datos de prueba para obtener los modelos. En efecto, aplicando la función directamente sobre el conjunto de prueba, se podría utilizar los otros casos de prueba para encontrar los 10 vecinos más cercanos que podrían ser usados para llenar cada valor faltante. Aunque no se usará la información sobre las variables

objetivo, la cual sería realmente incorrecta, se puede evitar este proceso utilizando los datos de entrenamiento para encontrar los vecinos. Esto sería más correcto pero también más realista en el sentido de que si se fueran a aplicar los modelos en problemas reales, se obtendrían probablemente las muestras de agua secuencialmente, una a la vez. La **función knnImputation()** tiene un argumento extra que puede ser utilizado para las situaciones de llenar los valores faltantes en un conjunto de datos de prueba. Se puede utilizar la siguiente instrucción:

```
> clean.test.algae <- knnImputation(test.algae, k = 10, distData = algae[,
+ 1:11])
```

El argumento de **distData** nos lleva a suministrar un conjunto de datos extra donde los 10 vecinos más cercanos son encontrados para cada caso con valores faltantes en el data frame **test.algae**. Note que se ha omitido las variables objetivo del conjunto de datos **algae**, como el conjunto de prueba no incluye información sobre estas variables.

Con el siguiente código se obtendrá la matriz con las predicciones para el conjunto de prueba enttero:

```
> preds <- matrix(ncol=7,nrow=140)
> for(i in 1:nrow(clean.test.algae))
+   preds[i,] <- sapply(1:7,
+                       function(x)
+                         predict(bestModels[[x]],clean.test.algae[i,])
+                       )
```

Con este simple código se obtuvo una matriz (**preds**) con las 7 x 140 predicciones requeridas. En esta etapa se puede comparar estas predicciones con los valores reales, para obtener alguna realimentación (“feedback”) sobre la calidad de nuestro enfoque para el problema de la predicción. Los valores verdaderos del conjunto de datos de prueba están contenidos en el data frame **algae.sols**. El siguiente código calcula los valores de NMSE de los modelos:

```
> avg.preds <- apply(algae[,12:18],2,mean)
> apply( ((algae.sols-preds)^2), 2,mean) /
+ apply( (scale(algae.sols,avg.preds,F)^2),2,mean)
```

	a1	a2	a3	a4	a5	a6	a7
	0.4650380	0.8743948	0.7798143	0.7329075	0.7308526	0.8281238	1.0000000

Primero se obtienen las predicciones de la línea base del modelo utilizado para calcular el NMSE, el cual, en este caso consiste en predecir el valor promedio de la variable objetivo. Después se procede a calcular los valores de NMSE para los siete modelos/alga. La función **scale()** puede ser utilizada para normalizar el conjunto de datos. Este trabaja restando el segundo argumento del primero y después divide el resultado entre el tercero, a menos que el argumento sea falso (FALSE), como es el caso anterior. En este ejemplo se está utilizando para restar un vector (el valor objetivo promedio de todas las siete algas) de cada línea de una matriz.

Los resultados obtenidos previamente con la estimación de la validación cruzada confirman la dificultad en obtener un buen resultado para el tipo de alga 7, mientras que para los otros problemas los resultados son más competitivos, en particular para el tipo de alga 1.

En resumen, con la selección del modelo apropiado, estamos capacitados para obtener valores interesantes para estos problemas de predicción.